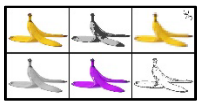
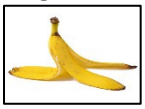
William McRoberts

Completed



Original



```
from jes4py import *
# William McRoberts - 10/21/24
# Main function
def collage():
    pic = makePicture(getMediaPath("banana.jpg"))
    # Add the path to your signature image
    signature = makePicture(getMediaPath("signature.png"))
    collage = makeEmptyPicture(929, 447, black)

    # Edit pictures
    p0 = scale(pic, .6)
    p1 = scale(pic, .6)
    p1 = poster(p1)
    p2 = scale(pic, .6)
    p2 = grayScale(p2)
    p3 = scale(pic, .6)
    p3 = colorSwap(p3)
    p4 = scale(pic, .6)
    p4 = edgeDetect(p4)
    p5 = scale(pic, .6)
    p5 = mosaic(p5)

    signature = scale(signature, .3)
    signature = sign(signature, black)

    width = getWidth(p0)
    height = getHeight(p0)

    # Overlay onto canvas
    overlay(collage, p0, 5, 5)
    overlay(collage, p1, 10 + width, 5)
    overlay(collage, p2, 5, 10 + height)
    overlay(collage, p3, 10 + width, 10 + height)
    overlay(collage, p4, 15 + (2 * width), 10 + height)
    overlay(collage, p5, 15 + (2 * width), 5)
    overlay(collage, signature, getWidth(collage) - getWidth(signature) - 5, 5)

    show(collage)

# Scales original picture by the given factor
def scale(source_picture, factor):
    new_width = int(getWidth(source_picture) * factor)
    new_height = int(getHeight(source_picture) * factor)
    new_picture = makeEmptyPicture(new_width, new_height)
    source_x = 0
    for new_x in range(0, getWidth(new_picture)):
        source_y = 0
        for new_y in range(0, getHeight(new_picture)):
            color = getColor(getPixel(source_picture, int(source_x), int(source_y)))
            setColor(getPixel(new_picture, new_x, new_y), color)
            source_y = source_y + (1 / factor)
```

```
        source_x = source_x + (1 / factor)
    return new_picture

# Poster effect
def poster(picture):
    for pixel in getPixels(picture):
        red_value = getRed(pixel)
        green_value = getGreen(pixel)
        blue_value = getBlue(pixel)
        luminance = (red_value + green_value + blue_value) // 3
        if luminance <= 50:
            setColor(pixel, black)
        elif luminance > 50 and luminance <= 150:
            setColor(pixel, darkGray)
        elif luminance > 151 and luminance <= 200:
            setColor(pixel, lightGray)
        else:
            setColor(pixel, white)
    return picture

# Gray scale effect
def grayScale(picture):
    for pixel in getPixels(picture):
        new_red = getRed(pixel) * 0.299
        new_green = getGreen(pixel) * 0.587
        new_blue = getBlue(pixel) * 0.114
        intensity = new_red + new_green + new_blue
        setColor(pixel, makeColor(intensity, intensity, intensity))
    return picture

# Color swap effect
def colorSwap(picture):
    for pixel in getPixels(picture):
        new_red = getGreen(pixel)
        new_green = getBlue(pixel)
        new_blue = getRed(pixel)
        setColor(pixel, makeColor(new_red, new_green, new_blue))
    return picture

# Helper function to edgeDetect
def lum(pixel):
    redVal = getRed(pixel)
    greenVal = getGreen(pixel)
    blueVal = getBlue(pixel)
    return (redVal + greenVal + blueVal) // 3

# Edge detection/sketch effect
def edgeDetect(picture):
    for pixel in getPixels(picture):
        x = getX(pixel)
        y = getY(pixel)
        if y < getHeight(picture) - 1 and x < getWidth(picture) - 1:
            botRightPix = getPixel(picture, x + 1, y + 1)
            thisLum = lum(pixel)
            botRightLum = lum(botRightPix)
            if abs(botRightLum - thisLum) > 10:
                setColor(pixel, darkGray)
            else:
                setColor(pixel, white)
    return picture
```

```
# Overlays the new edited picture into the empty picture
def overlay(collage, pic, startX, startY):
    for x in range(getWidth(pic)):
        for y in range(getHeight(pic)):
            color = getColor(getPixel(pic, x, y))
            setColor(getPixel(collage, startX + x, startY + y), color)

# Mosaic effect/blur/lower resolution
def mosaic(picture):
    for x in range(0, getWidth(picture), 7):
        for y in range(0, getHeight(picture), 7):
            avg_color = average_color(picture, x, y, 7)
            set_block_color(picture, x, y, 7, avg_color)
    return picture

# Helper function to mosaic
def average_color(picture, start_x, start_y, block_size):
    red_total = 0
    green_total = 0
    blue_total = 0
    count = 0
    for x in range(start_x, min(start_x + block_size, getWidth(picture))):
        for y in range(start_y, min(start_y + block_size, getHeight(picture))):
            pixel = getPixel(picture, x, y)
            red_total += getRed(pixel)
            green_total += getGreen(pixel)
            blue_total += getBlue(pixel)
            count += 1
    return makeColor(red_total // count, green_total // count, blue_total // count)

# Helper function to mosaic
def set_block_color(picture, start_x, start_y, block_size, color):
    for x in range(start_x, min(start_x + block_size, getWidth(picture))):
        for y in range(start_y, min(start_y + block_size, getHeight(picture))):
            setColor(getPixel(picture, x, y), color)

# Reverse Chroma Key effect
def sign(picture, bg_color):
    for pixel in getPixels(picture):
        if getColor(pixel) != bg_color:
            setColor(pixel, white)
        else:
            setColor(pixel, black)
    return picture
```